RECOMP II USERS' PROGRAM NO. 1033

PROGRAM TITLE: SIGNAL CORPS RECOMP ASSEMBLY PROGRAM - SCRAP II

PROGRAM CLASSIFICATION: Executive and Control

AUTHOR: T. J. Tobias
U. S. Army Signal Engineering Agency
Arlington Hall Station
Arlington, Virginia

PURPOSE: SCRAP is an assembly program for the RECOMP II computer. It is designed to use mnemonic operation codes and symbolic, absolute, operand, or relative addresses. SCRAP uses all of the RECOMP II commands, as well as pseudo-operation codes and macro-instructions. The SCRAP processor requires two passes to complete the assembly of a program written in the SCRAP language. The primary working media of the SCRAP processor during assembly is paper tape for both input and output data. Printed output is optional during both the first and second passes of assembly.

DATE: January 1960

Published by

RECOMP Users' Library

at

AUTONETICS INDUSTRIAL PRODUCTS

A DIVISION OF NORTH AMERICAN AVIATION, INC.
3584 Wilshire Blvd., Los Angeles 5, Calif.

SIGNAL CORPS RECOMP ASSEMBLY PROGRAM

SCRAP II

by

T. J. TOBIAS, U. S. ARMY SIGNAL ENGINEERING AGENCY

INTRODUCTION: SCRAP is an assembly program for the RECOMP II computer. It is designed to use mnemonic operation codes and symbolic, absolute, operand, or relative addresses. SCRAP uses all of the RECOMP II commands, as well as pseudo-operation codes and macro-instructions. The SCRAP processor requires two passes to complete the assembly of a program written in the SCRAP language. The primary working media of the SCRAP processor during assembly is paper tape for both input and output data. Printed output is optional during both the first and second passes of assembly.

DESCRIPTION:

1. The SCRAP processor allows for an extremely flexible instruction format. Each line of SCRAP programming may have a symbolic location of up to eight alphabetic characters, a command, and an address of any one of six (6) types. For example, if a floating point number in location 1052 is to be multiplied by the decimal number, -71.394, and the result stored in a data area, this program might be written as follows:

LOCATION	COMMAND	T	ADDRESS	REMARKS
	FCA	N	1052	C(1052,3) to A,R
	FMP	F	-71.394	C(A,R) x (-71.394)
	FST	S	DATA	C(A,R) to DATA

These three instructions generally illustrate the major type of 'address' permitted in SCRAP; that is, absolute (N) addresses, symbolic (S) addresses, and operand (F,D,C,A) addresses. As may be noted, four (4) types of operand addresses are allowed. These will provide a means of entry for almost all types of data encountered in the programming of the RECOMP II computer. The permissible types of operand addresses are:

F - - Floating Point Numbers

D - - Fixed Point Numbers

C - - Command Format Data

A - - Alphabetic (Baudot) Data

As an illustration of programming using SCRAP, following is a subroutine for the assemoly of a floating point number where the C(A) is the integer part at 39 and the C(R) is the fractional part at 0.

LOCATION	COMMAND	T	ADDRESS	REMARKS
FIXTOFLO	SAX	S	TEMPSTO	C(A) to TEMPSTO
	ADD	D	+1+39	Return address +1 @ 39
	STA	S	EXIT	Store to exit line
	XAR	N	0	C(R) to A
	STO	S	TEMPSTO+1	C(A) to TEMPSTO+1
	FCA	S	CONSTANT	+39 @ 39 to R
	CLA	S	TEMPSTO	Integer Part to A
	FNM	N	0	Integer Part to Normalized FP
	FAD	S	TEMPSTO+1	Integer + Fractional Part to FP
	SR			Set Next Instruction Right
EXIT	TRA	N	0	Exit Line
TEMPSTO	DECIMAL	D	+0+0	TS for integer part
	DECIMAL	D	+0+0	TS for fractional part
CONSTANT	COMMAND*	C	+0000000-0000000	Exponent of zero for fractional part
	DECIMAL	D	+39+39	Constant of +39 @ 39

*Zero in command format is to illustrate the use of the C type address in this example.

This program contains many of the permissible types of instructions which may be used when programming in the SCRAP language. Of particular note in the example is the use of the relative address, 'TEMPSTO+1', which

refers to the location one (1) word beyond the location named 'TEMPSTO'. Relative addressing of both full and half words from 00001 to 77771 is permitted relative to any symbolic address.

2. The four fields of data of a SCRAP program, LOCATION, COMMAND, T, and ADDRESS, may contain the following information:

a. LOCATION: The location may contain a symbolic 'tag' to identify a memory location. This field may contain from one (1) to eight (8) alphabetic characters (A-Z). The typewriter functions, figures shift, carriage return, tab, and blank, 02, are not allowed. The letter 'C' should not be used as a location symbol since this character is reserved as a special address symbol. The location field may be blank.

b. COMMAND: The command field must contain a mnemonic operation code, pseudo-operation code, or macro-instruction which is recognized by SCRAP. In addition to the RECOMP II operation codes the following are permissible:

COMMAND	DEFINITION
HALT	same as HTR
HLT	same as HTR
DISC	DIS - Command format
DISD	DIS - BCD format
TYWC	TYW - Command format
TYWD	TYW - BCD format
PNWC	PNW - Command format
PNWD	PNW - BCD format
PTWC	PTW - Command format
PTWD	PTW - BCD format

Note: DIS, TYW, PNW, and PTW will be assembled as command format. The complete list pseudo-operation codes is described in paragraph 3, and the macro-operation codes are discussed in paragraph 4.

c. TYPE OF ADDRESS FIELD, T: This code specifies the type of address used in the address field. The following six codes are used:

TYPE CODE	TYPE OF ADDRESS
S	Symbolic
N	Absolute Numeric Address, Octal and half word bit
D	Fixed point decimal
F	Floating point decimal
C	Command format
A	Alphabetic

The type of address field may be blank; if it is, the address will be interpreted as symbolic.

d. ADDRESS: The address field may contain any one of the six types of data identified above. The required format of these data is as follows:

(1) SYMBOLIC: A symbolic address may contain from one (1) to eight (8) alphabetic characters (A-Z). The typewriter functions figures shift, carriage return, tab, and blank, 02, are not permitted. A symbolic address may in addition have an increment or decrement applied to it at assembly time. The increment or decrement must have the same form as an absolute address. For example:

ADDRESS	REMARKS
EXIT	A reference to the location named 'EXIT'
TEMPSTO	A reference to the location named 'TEMPSTO'
TEMPSTO+1	The location one word beyond 'TEMPSTO'
EXIT-00001	One half word back from 'EXIT'
AREA-00021	Two and one half words back from 'AREA'

The special address 'C' may be used to refer to locations relative to the present instruction.

For example:

LOCATION	COMMAND	T	ADDRESS	REMARKS
	CLA		DATA	C(DATA) to A
	TPL		C+1	Are C(DATA) plus?
	TRA		OUT	C(DATA) are not plus
	TZE		C+1	Are C(DATA) both plus and zero?
	TRA		OUT	C(DATA) are plus but not zero
	ANYOP			C(DATA) are plus and zero

This sequence of instructions illustrates a comparison operation to determine if the contents of location DATA are plus zero. The use of the special address form 'C' allows for relative addressing to the present instruction and in this case eliminates the need to write the two location 'tags' which would have been required without this 'self-relative' feature.

(2) ABSOLUTE NUMERIC ADDRESS: An absolute numeric address field may contain from one (1) to five (5) numeric characters. The first four characters of the field must be an octal address. The last character of the field represents the half word bit as in the normal RECOMP command format. If the field is less than five (5) characters, it will represent an octal address, right justified, and with the half word bit of zero. For example:

ABSOLUTE ADDRESS	EQUIVALENT COMMAND FORMAT
7760	7760.0
3	0003.0
745	0745.0
57021	5702.1
00001	0000.1
0	0000.0

(3) FIXED POINT DECIMAL: A fixed point decimal number used in the address field may contain a maximum of sixteen (16) characters including the sign, decimal point, and location of the binary point. In addition, neither the integer nor the fractional part of the number may contain more than eleven (11) characters. The location of the binary point is specified by the use of the suffix ±BB. The general form of a fixed point decimal number is ±IIII.FFFF±BB. The leading sign may be omitted if the number is positive. If the location of the binary point is omitted, however, the number will be converted as a floating point decimal number but will only be allocated one word of storage. There are no error halts for this situation. Examples of fixed point decimal numbers are as follows:

ADDRESS	MEANING	RESULTANT COMMAND FORMAT
+1+39	+1 @ 39	+0000000-0000001
1+20	+1 @ 20	+0000000+0000000
-100+38	-100 @ 38	-0000000-0001440
-.1-3	-0.1 @ -3	-6314630-6314630
0.125+0	+0.125 @ 0	+1000000-0000000

If the specified binary point would cause the loss of significant (left hand) bits, the number will not be converted as specified. There will be no error halt or error indication.

(4) FLOATING POINT DECIMAL: A floating point decimal number used in the address field may contain a maximum of sixteen (16) characters including sign and decimal point. Neither the integer part of the number nor the fractional part may contain in excess of eleven (11) characters. The sign may be omitted if the number is positive. Examples of floating point decimal addresses are as follows:

ADDRESS	MEANING	RESULTANT COMMAND FORMAT
+1	+1	+4000000-0000000
		+0000000-0000001

ADDRESS	MEANING	RESULTANT COMMAND FORMAT
-0.25	-0.25	-4000000-0000000 -0000000-0000001
100	+100	+6200000-0000000 +0000000-0000031
-10.75	-10.75	-5300000-0000000 +0000000-0000020

(5) COMMAND: A word of command format data may be used in the address field. It will be assembled exactly as specified. For example:

ADDRESS	RESULTANT COMMAND FORMAT
+7766010-1234561	+7766010-1234561
-0000000+7777400	-0000000+7777400

If a non-octal numeric character is entered as command data, it will be assembled in the command limited to its three low order bits. That is, eight would be assembled as zero and nine as one. There are no error halts or other indications of this condition.

(6) ALPHABETIC: An address field may contain from zero (0) to eight (8) alphabetic characters. These will be assembled in the baudot code form with the data right justified. The characters carriage return, tab, and blank, 02, are not permitted. Examples of alphabetic data are as follows:

ADDRESS	RESULTANT COMMAND FORMAT
AREA	-0000000-0520211
Z	-0000000-0000101
ANYPLACE	-1545531+1033401

3. SCRAP contains a number pseudo-operation codes which cause certain functions to be performed by the SCRAP processor at assembly time. These functions may also cause some object code to be produced.

The following is the SCRAP repertoire of pseudo-ops.

PSEUDO-OPERATION	DEFINITION
ORG	ORIGIN: This pseudo-operation code will cause the next instruction to be assembled in the location specified by the address part of the ORG pseudo-operation. The address of this command must be numeric. For example:

LOCATION	COMMAND	T	ADDRESS
	ORG	N	500

This will cause the next instruction to be assembled in 0500.0

DEF — DEFINITION: This command specifies that the symbolic location in the LOCATION field is defined to be the absolute address specified in the address field. For example:

LOCATION	COMMAND	T	ADDRESS
START	DEF	N	1200
L	DEF	N	7760
DATA	DEF	N	5000

The LOCATION field must be symbolic and the ADDRESS field must be a numeric absolute address.

EQU — EQUIVALENCE: This pseudo-operation code identifies two symbols as being equivalent. The two symbols so identified may be used interchangeably. For example:

LOCATION	COMMAND	T	ADDRESS
DATAAREA	EQU	S	DATA
TS	EQU	S	TEMPSTO

The symbol in the LOCATION field should be identified by the equivalence pseudo-operation before it is used in the program or it may not be recognized and assembled correctly.

SL — SET LEFT: This command will cause the next instruction to be assembled in the left hand side of the word. This command may cause an ARS 0000.0 to be assembled as a dummy instruction if required to cause the set left operation.

SR — SET RIGHT: This command will cause the next instruction to be assembled in the right hand side of the word. As in the case of the SL pseudo-operation code, this may cause a dummy of ARS 0000.0 to be generated.

SB — SET BLOCK: This command will cause the next instruction to be assembled in the left side of the next modulo eight word. Such dummy instructions as may be generated will be ARS 0000.0

END — This pseudo-operation code signifies the end of the data to be assembled. No other operation may follow the END pseudo-operation code.

PAUSE — This operation code will cause the SCRAP processor to stop assembling. Restart is accomplished by depressing the start button.

ALPHA or ALF — ALPHABETIC DATA: This pseudo-operation code indicates that the address field contains alphabetic data. The type code must be A.

DECIMAL or DEC — DECIMAL DATA: This command indicates that the address field contains decimal information. The type code must be either an F or a D.

COMMAND or COM — COMMAND DATA: This pseudo-operation code indicates that the command field contains command format data. The type code must be C.

4. SCRAP II also allows for the use of macro-instructions. These may cause the production of several lines of coding for each macro that is given. The macros are used in the same manner as normal operation codes. If the macro has several arguments these are listed in successive address fields. For example:

LOCATION	COMMAND	T	ADDRESS	
anytag	TNZ		any permissible	Operation code Transfer on non-zero

LOCATION	COMMAND	T	ADDRESS	
anytag	ZMT		any permissible	Operation code Zero mode transfer
anytag	SAM		no. argument	Operation code Set A Register minus

The detailed discussion of the construction of macro instruction is contained under the operating procedures for macros.

5. The SCRAP processor requires two passes to complete the assembly of the program. During the first pass an assignment table is accumulated in which all the symbolic references are listed. If the symbol has also been used as a location tag then an absolute assignment for that symbol is stored. If a symbolic address is not also used in the program as a location, no specific assignment may be made. These symbols will be referred to as unassigned symbols. Also during the first pass a table of equivalences is constructed for use during both the first and second passes. During the first pass all operand addresses are replaced by names. That is, the first fixed point constant encountered in the program is assigned the name 'FIXCN01'. This name is also substituted on the output tape for this constant. The floating point constants are assigned the name 'FLOCNnn', the alphabetic constants the name 'ALFCNnn', and the command format constants the name 'COMCNnn'. Each different constant is only named once and becomes a part of a constant pool. The macro instructions are also expanded to their full representation. At the end of the first pass all of the constants used as operand addresses are assigned locations immediately after the end of the program locations. The assignment table is then printed out. All unassigned symbols may be assigned absolute locations at this time (during the printing operation) or the assignment table may be printed a second time with assignment of all unassigned symbols taking place at that time. This feature of assignment of unassigned symbols or non-assignment at the programmer's option allows for checking of mispelling, omissions, or other errors which may not be obvious if all unassigned symbols are assigned. A good procedure is to obtain a listing of the assignment table first without assignment of the unassigned symbols, and then a second listing with assignment if desired. A listing of all operand addresses and the corresponding names is also printed out for cross reference and checking purposes.

6. During the second pass all instructions and data are converted to the correct command format. The assignment table is used to obtain the corresponding absolute address for all symbolic addresses. Absolute addresses and data are converted to the proper command format words. An output listing of the assembly is optional during this pass. The object program is punched into paper tape in command format.

7. Insertions and deletions may be made during the first pass. This is accomplished by use of a preset stop to the beginning of the first pass program and by inputing the necessary corrections, additions,

or deletions from the typewriter. This allows for the reassembly of a program, requiring minor changes, without having to re-keypunch the entire input tape.

8. At the end of either the first or second pass a copy of all significant tables and other data may be obtained on paper tape. This will allow for the continuation of assembly at some later time beginning at the point where the previous program ended, or will allow the first and second passes of assembly to be accomplished on a non-continuous basis.

SUMMARY: The SCRAP processor provides for an extremely flexible instruction format, including provisions for symbolic, absolute, operand, and relative addresses. The SCRAP processor also provides for corrections, insertions, and deletions during the first pass of assembly and for assembly of a program in sections. This assembly program also provides for the use of macro-instructions which may, if desired, be constructed for only one time use. These features make the SCRAP assembly program an extremely flexible aid to programming and provides a base for even more complex automatic programming systems.

SIGNAL CORPS RECOMP ASSEMBLY PROGRAM, SCRAP II

APPENDIX I

SCRAP OPERATING INSTRUCTIONS

1. GENERAL: The SCRAP processor has several modes of operation and also provides for a number of options during (or after) processing. These operating procedures contain information regarding the following:

a. "Key Punching" paper tape with SCRAP.

b. Procedures for the First Pass.

c. Procedure for the Second Pass.

d. Insertions, Deletions, and Corrections during the First Pass.

e. Dumping of the Assignment Table and other Data.

f. Construction of Macro-Instructions.

g. Restrictions and Program Halts.

2. KEY-PUNCHING SCRAP INPUT TAPE: The SCRAP program may be used to process input from the typewriter and produce a paper tape in the proper format for later assembly. The SCRAP processor reads and edits the input information from typewriter, performs a cursory check for errors, and produces the properly formated paper tape.

a. Key Punching

(1) Clear locations 0500-4277 to negative zero.

(2) Load SCRAP II Program.

(3) Set sense switch B and C on; Sense D off.

(4) Set typewriter margin at 10; tabs at 20, 29, and 32.

(5) Depress START 1 to begin.

(6) Type command field and tab (or tab if blank).

(7) Type command field and tab.

(8) Type "T" field and tab (or tab if blank).

(9) Type address field and a carriage return. The output paper tape will be punched at the completion of the carriage return.

(10) Repeat steps 6 through 9.

(11) If an error is made and detected before the carriage return at the end of the line, it may be deleted by depressing the blank key immediately to the right of the "M" Key. The line is then retyped.

(12) No error may be corrected after the carriage return as the data is already on tape.

(13) If, as a result of typing too quickly, an output error is caused, this condition may be corrected by:

(a) Depressing error reset; then

(b) Depressing START 1, and then

(c) Depressing the blank key next to the "M" Key. The line may now be retyped by following steps 6 through 9.

b. Termination Key Punching

(1) If the last entry on tape is the pseudo-operation END, the tape may be terminated by:

(a) Tabbing blank fields until the end of group termination occurs (Output is grouped on tape 16 lines of coding per group).

(b) Or, by setting the punch to manual and punching 9 blanks (00), L 77000, carriage return, 3 blanks (00), and an "S".

(2) If the program is to be key-punched in sections, each section may be terminated as follows:

(a) Type a line of coding with the pseudo-op PAUSE.

(b) Tab at least one blank line of coding (no command or T field data).

(c) Terminate by either one of methods outlined in (1) above.

(d) Clear location 5013 to +zero before restarting.

It should be noted that the PAUSE line is not essential but is a useful means of stopping assembly while the next section of paper tape is placed in the photo reader. The blank line is necessary in that the absence of a command field causes the reading of a new group from paper

tape. During assembly the PAUSE causes a temporary stop and after the loading of the photo reader, assembly is restarted by depressing the start button.

3. FIRST PASS: The first pass of the SCRAP assembly allows for optional input and output methods as well as the optional assignment of unassigned symbols. Insertions, corrections, and deletions may also be made during the first pass. These changes are discussed in greater detail in paragraph 5 below. The setting of the sense switches determines the choice of options in the first pass as follows:

		B	C
a.	Paper tape Input Only.	off	off
b.	Typewriter Input Only	off	on
c.	Paper tape Input and Typewriter Output.	on	off
d.	Key Punch only (No Assembly)	on	on

Paper tape output occurs for all forms of input. Option (d) above is included in the list of sense switch settings for comparison purposes only since no assembly occurs when using this option. The assignment of unassigned symbols will occur only after the pseudo-operation END and only if sense D is on. In practice it may be preferable to obtain a copy of the unmodified assignment table first, and then to exercise the assignment of unassigned symbols option. The first pass operating procedure is as follows:

(1) Clear locations 0500-4277 to negative zero.

(2) Load SCRAP program.

(3) Set sense switches, typewriter margin and tabs and load tape into photo reader.

(4) Depress START 1 to begin assembly.

(5) When the pseudo-op END occurs the SCRAP processor will perform the following actions:

  (a) Print a list of all constants, then

  (b) Print the assignment table

  (c) Print END FIRST PASS and halt.

(6) A second copy of the assignment table may be obtained by depressing the start button (or starting at 4513). The position of Sense D may be changed if desired.

4. SECOND PASS: The second pass completes the assembly operation. The input to this pass is the output tape from the first pass. The output from the second pass is a copy of the object program on paper tape in command format. The Second Pass Procedure is as follows:

a. Load SCRAP program and assignment table (this is necessary only if the first and second passes have not been run continuously).

b. Load photo reader with output tape from the first pass.

c. Set typewriter tabs and sense switches (Sense C off and B either on or off.)

d. Depress START 2 to begin.

e. Assembly will proceed until the END pseudo-op occurs.

5. INSERTIONS, DELETIONS, AND CORRECTIONS: Changes may be made to the program during the first pass of original assembly or during a reassembly of the program. It requires that the input media be punched paper tape and that the optional printed listing of assembly be allowed (at least partly). Procedures for these changes are as follows:

a. Set Sense Switch B on and C off.

b. Set Read Out Knobs to location 4600.

c. For a deletion:

(1) Set Preset Stop to 4600.1.

(2) After the line to be deleted has been printed and the computer stops, depress START 1 to cause a deletion. Repeat if necessary.

(3) Set Preset Stop to neutral and depress START to resume assembly.

d. For an Insertion:

(1) Set preset stop to 4600.0.

(2) When the computer stops at the point of the insertion;

(a) Set preset stop to neutral.

(b) Set sense switch B off and C on.

(c) Depress START.

(3) Type insertion(s) from typewriter.

(4) Terminate the insertion operation by a PAUSE pseudo-operation.

(5) Set sense switch B on and C off and then depress START 1 to resume assembly.

e. For a Correction:

(1) Set Preset Stop to 4600.1.

(2) After the line to be corrected is printed and the computer stops;

(a) Set Sense Switch B off and C on.

(b) Set Preset Stop to neutral.

(c) Depress START 1.

(3) Type correction from typewriter.

(4) Terminate the correction operation by use of the PAUSE pseudo-operation.

(5) Set Sense Switch B on and C off and then Depress START 1 to resume assembly.

f. Changes may also be made to the assignment table at the end of the first pass. For example, a mispelling might cause the following entries in the printout of the assignment table:

SCAN B	+0000000-0035620
SYMBLOIC	-0000000-0000000
GETPAREN	+0000000-0036120
SYMBOLIC	+0000000-0036000
GETITEM	+0000000-0036701

The second entry above, "SYMBLOIC", is the result of mis-pelling the word "SYMBOLIC". Since the two are equivalent names, the command format word, +0000000-0036000, may be entered in place of the - zero word. The assignment table

begins in location 0500 and contains two word items for each symbol, the first word being the name of the symbol and the second word the assignment. Thus, a manual search of memory will uncover the location of the particular unassigned symbol and the correction may be entered from the console. This procedure will save reassembly time for relatively minor errors of mispelling or omission.

6. DUMPING OF ASSIGNMENT TABLE: The assignment table and other significant data may be dumped on paper tape by use of a SAVE program. This will allow sectional processing of the first pass or interrupted processing of the first and second passes. The SAVE program is started by use of the START 3 button. Restart of assembly may be accomplished at some later time by filling the SAVE tape after loading the SCRAP program tape, thus restoring the program to its previous state.

7. CONSTRUCTION OF MACRO-INSTRUCTIONS: Macro-instructions may be added to the SCRAP II list of operation codes by adding an appropriate definition of the macro to the SCRAP II processor. In general, a macro consists of a name and a list of arguments as follows:

```
tag   MACRONAME   A#1
                  A#2
                  A#3
                   :
                  A#n
```

This generates a list of instructions of the following form:

```
tag   MACHINECODE   X#n1
      MACHINECODE   X#n2
                     :
      MACHINECODE   X#nm
```

Where, the X#n's are either from the list of arguments A#n or from the list of possible machine (assembly) addresses. The macro, transfer on non-zero, TNZ, might be written as follows:

```
MACRO:        tag   TNZ   A#1; and produce
```

OBJECT CODE:

```
        tag   TZE   C+1
              TRA   A#1
```

This would generate the following code for various definitions of A#1;

```
      TNZ     N     7770     would generate      TZE          C+1
                                                 TRA     N    7770

TEST  TNZ             NODATA   would generate TEST TZE          C+1
                                                 TRA          NODATA
```

b. The interpretation of the macro-instruction depends upon the definition coding of the macro. This skeleton coding is entered into the SCRAP II processor in the format used for instructions in SCRAP. In addition, a new instruction form for the Arguments (A#n's) is added for use in the skeleton coding (This address form never appears externally from the processor). In addition to the definition (skeleton coding) of the macro, the name and limits of the macro must be added to the list of operation codes of SCRAP. A macro-instruction, therefore, must be added to the list of operation codes and the definition coding of the macro must be available to the SCRAP II processor.

c. The operation code table entry must have the following form:

```
First Word          MACRONAME          (7 character maximum)

Second Word         +aa67100+ppssss0   (command format)

    where,   aa     number of arguments

             pp     number of resulting instructions

             ssss   location of definition
```

The entry requires two words. The first word is in alphabetic form and contains the name of the macro. The second word defines the limits of the macro and the location of macro definition. Locations 0330-0475 are the available for op table entries referring to macros. This table must be extended continuously and the two words after the last entry must be negative zero.

d. The macro definition coding consists of three word entries in the following form:

First Word	AbbbbRRR
Second Word	( )
	(address)
Third Word	( )

where, A is an appropriate code corresponding, in function, to the type of address code; and, RRR is a RECOMP II operation code. (b is used here to represent a blank, 00).

The six SCRAP II address forms are permitted in the address part, as well as a special address form for the argument numbers. The format for macro definition coding is as follows:

TYPE OF ADDRESS	CODE A	ADDRESS FORMAT
Symbolic	B	SYMBOLIC * bbbbbbbb
Numeric	E	±bbnnnnn bbbbbbbb
Command	H	±ccccccc ±ccccccc
Fixed Point Decimal	O	{number}
Floating Point Decimal	D	{number}
Alphabetic	L	AAAAAAAA bbbbbbbb
Argument	b	+5400000-00000n0 bbbbbbbb

*This second word may be an increment or decrement of the form ±bbnnnnn.

Locations 7000 to 7577 are available for storage of macro definitions. This allows for a maximum of 192 lines of macro coding definitions.

e. The TNZ macro could be written in the following form:

MACRO			OBJECT CODE		
tag	TNZ	A#1	tag	TZE	C+1
				TRA	A#1

LOCATION	COMMAND	T	ADDRESS	REMARKS
	ORG	N	0030	
	ALF	A	TNZ	MACRONAME
	COM	C	+0167100+02700000	OP TABLE CODE WORD
	ORG	N	7000	
	ALF	A	TZE	
	ALF	A	C	TZE C+1
	ALF	A	+bbbbblb	
	ALF	A	TRA	
	COM	C	+5400000-0000010	TRA A#1
	ALF	A	———	
	END			

This definition of the macro TNZ may be keypunched using SCRAP; and, in the above form (only data (ALF, DEC, or COM) or location, ORG, pseudops), it may be translated to an object tape by using pass two only of SCRAP.

f. A macro to set the A register minus, SAM, might be written as follows:

MACRO			OBJECT CODE		
tag	SAM		tag	EXT	C-7777771+7777771

LOCATION	COMMAND	T	ADDRESS	REMARKS
	ORG	N	0332	
	ALF	A	SAM	MACRONAME
	COM	C	+0167100+0170060	OP TABLE CODE WORD
	ORG	N	7006	
	AEF	A	HbbbbEXT	EXT C-77777...
	ALF	A	-7777771	
	ALF	A	+7777771	

g. A macro to move a block (8 words) to the L loop and to transfer to 7760 (ZMT) might be written as follows:

MACRO			OBJECT CODE		
tag	ZMT	A#1	TAG	CTL	A#1
				TRA	7760
				SB	

LOCATION	COMMAND	T	ADDRESS	REMARKS
	ORG	N	0334	
	ALF	A	ZMT	MACRONAME
	COM	C	+0167100+0370110	OP TABLE CODE WORD
	ORG	N	7711	
	ALF	A	CTL	CTL A#1
	COM	C	+5400000-0000010	
	ALF	A	-------	
	ALF	A	EbbbbTRA	TRA N 7760
	ALF	A	+bb7760b	
	ALF	A	-------	

LOCATION	COMMAND	T	ADDRESS	REMARKS
	ALF	A	SB	} SB
	ALF	A	_____	
	ALF	A	_____	

h. A macro to increment a counter (COUNT) might be written as follows:

	MACRO			OBJECT CODE	
tag	COUNT	A#1	tag	CLA	A#1
		A#2		ADD	A#2
				STO	A#1

LOCATION	COMMAND	T	ADDRESS	REMARKS
	ORG	N	0336	
	ALF	A	COUNT	MACRONAME
	COM	C	+0267100+0370220	OP TABLE CODE WORD
	ORG	N	7022	
	ALF	A	CLA	} CLA A#1
	COM	C	+5400000-0000010	
	ALF	A	_____	
	ALF	A	ADD	} ADD A#2
	COM	C	+5400000-0000020	
	ALF	A	_____	
	ALF	A	STO	} STO A#1
	COM	C	+5400000-0000010	
	ALF	A	_____	

i. The macro instructions written for SCRAP II must observe the following restrictions:

(1) No macro may use another macro in its definition coding.

(2) A macro may have a maximum of 12 arguments.

(3) A macro must produce at least one line of output coding.

(4) If a macro may have a location tag, the definition coding may not begin with SR, SL, SB, PAUSE, or ORG.

j. If it is desired, pseudo-ops may be added to SCRAP II which will cause a minus op code to be produced in the object code. These will require only an entry in the op code table in one of the following forms:

(1) First Word OPNAME

Second Word +0046360±cc65120

In this form the half word bit will be preserved as is required for TRA, TZE, STA, etc.

(2) First Word OPNAME

Second Word +0046360±cc65060

In this form the half word bit will always be set to zero.

(3) First Word OPNAME

Second Word +0046360±cc65160

In this form the half word bit will always be set to one. Where, ±cc is the minus op code. For example, if a long right shift, LRS, is to be added to the repertoire with an op code of -40. This could be written as follows (in SCRAP notation):

```
ORG     N     0330
ALF     A     LRS
COM     C     +0046360-4065060
```

(Assuming locations 0330-0331 are available in the op table)
The address parts of the minus op codes will be handled in exactly the same manner as a normal RECOMP II op code. A minus op which requires a full word may need to be prefaced with a SL pseudo-op. Also, the configuration of the "address" of the minus op must necessarily correspond to at least one of six of the permissible SCRAP II address forms. The name given to the op code may not exceed seven characters. The name may include a figures shift. The name of a pseudo-op or macro may, therefore, be any one of the following forms:

TNZ

SET3

7X2

ARS+

-00

Thus, the pseudo-op named -00 could be defined to have the absolute value of -00. The value assigned to any op code is determined solely by the entry in the op code table

8. RESTRICTIONS AND PROGRAMMED HALTS:

a. The following are the restrictions referring to the number items. Each of these has a related Error Halt. These restrictions are as follows:

(1) Maximum of 512 Symbolic Names

(2) Maximum of 256 constants and no more than 99 of any one type (A, D, F and C).

(3) Maximum of 64 Equivalences.

b. The SCRAP II programmed halts are as follows:

LOCATION	INDICATION	ERROR CONDITION	CORRECTIVE ACTION
4334.0	(...01..)*	More than 512 Symbols	No immediate action; segment program

LOCATION	INDICATION	ERROR CONDITION	CORRECTIVE ACTION
7777.0	(...02..)	Search Error of Constant Pool	Clear memory? Restart at 4345.1 to try again.
7777.0	(...03..)	More than 256 Constants	No immediate action; segment program.
4423.1	(...04..)	Location Counter greater than 7757.1	No immediate action; Charge ORG or segment program.
7777.0	(...05..)	Search error of equivalence Table	Clear memory? Restart at 4452.1 to try again.
7777.0	(...06..)	More than 64 Equivalences	No immediate action; Reduce equivalences.
4600.0	"ILLEGAL OP CODE"	No find OP CODE	Deletion or correction action.
4720.0	None	More than 99 Constants	No immediate action; Restart at 4724.1 will ignore Constant but will cause Halt in Second Pass to 6545.1
6540.1	None	Location not in Assignment Table	To ignore, use START.
6545.1	None	Address not in Assignment Table	To ignore, use START (address of assembled instruction will be 0000.0).
0000.0	None	Possible paper tape read error	See following paragraph.

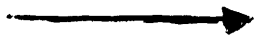
*Displayed on Console

c. The SCRAP II normal halts are as follows:

LOCATION	INDICATION	MEANING
4513.0	"END FIRST PASS"	Same
7777.0	Punching of Leader	End of Second Pass

LOCATION	INDICATION	MEANING
0000.0	None	End of SAVE

d. Paper Tape may be re-read if necessary by moving the tape back to the last gap and then restarting at 5311.0. The gap has the following punching:



Direction of tape movement

... data F bbbbbbbbbbb S bbb C/R 00077L bbb ...

Gap of 11 blanks

Such incomplete reads are occasionally caused by shiny spots on tape. A shiny spot which will cause a misread of a carriage return will cause a branch to zero (and a halt to zero). The cause of the halt may be checked by examining the tape in the reader to determine if it has stopped on a gap, if not a shiny spot may have caused the halt. The possibility of restart and re-read after blackening the shiny spot allows for the salvaging of the assembly operation. Further minor correction of the program may be necessary and may be accomplished by use of the correction, insertion, and deletion provisions detailed in paragraph 5.

SIGNAL CORPS RECOMP ASSEMBLY PROGRAM, SCRAP 11

APPENDIX 11

KEYPUNCHING

```
                ORG         N       1000
CUBEROOT        SAX                 X
                ADD         D       +1+39
                STA                 ERROR
                ADD         D       +1+39
                STA                 NORMAL
                XAR         N       0
                STO                 X+1
                FCA         F       +1.0
                FST                 RESULT
                SL
LOOP            FSQ                 X
ERROR           TOV         N       0
                FST                 X
                FSQ                 X
                FST                 X
                FMP                 RESULT
                FST                 RESULT
                FCA                 X
                FSB         F       +1.0
                EXT         C       -7777771+7777771
                FAD         F       +0.0000000001
                TMI                 LOOP
                SL
                FCA                 RESULT
NORMAL          TRA         N       0
RESULT          DECIMAL     F       -0
X               DECIMAL     F       -0
                END

L77000
$
```

APPENDIX II

FIRST PASS OF ASSEMBLY

```
LOCATION        COMMAND         ADDRESS

                ORG             +1000
CUBEROOT        SAX             X
                ADD             (+1+39)
                STA             ERROR
                ADD             (+1+39)
                STA             NORMAL
                XAR             +0
                STO             X+1
                FCA             (+1.0)
                FST             RESULT
                SL
LOOP            FSQ             X
ERROR           TOV             +0
                FST             X
                FSQ             X
                FST             X
                FMP             RESULT
                FST             RESULT
                FCA             X
                FSB             (+1.0)
                EXT             (-7777771+7777771)
                FAD             (+0.0000000001)
                TMI             LOOP
                SL
                FCA             RESULT
NORMAL          TRA             +0
RESULT          DECIMAL         (-0)
X               DECIMAL         (-0)
                END
FIXCNO1         +1+39
FLOCNO1         +1.0
COMCNO1         -7777771+7777771
FLOCNO2         +0.0000000001

CUBEROOT        +0000000-0010000
X               +0000000-0010160
ERROR           +0000000-0010051
NORMAL          +0000000-0010131
RESULT          +0000000-0010140
LOOP            +0000000-0010050
FIXCNO1         +0000000-0010200
FLOCNO1         +0000000-0010210
COMCNO1         +0000000-0010230
FLOCNO2         +0000000-0010240
ENDTABLE        +0000000-0010260

END FIRST PASS
```

APPENDIX II

SECOND PASS OF ASSEMBLY

LOCATION	COMMAND	ADDRESS	
	ORG	+1000	L10000
CUBEROOT	SAX	X	
	ADD	FIXCNO1	+1510160+0110200
	STA	ERROR	
	ADD	FIXCNO1	+4210051+0110200
	STA	NORMAL	
	XAR	±0	+4210131+4300000
	STO	X+1	
	FCA	FLOCNO1	+6010170+3010210
	FST	RESULT	
	SL		+3510140+4000000
LOOP	FSQ	X	
ERROR	TOV	+0	+4410160+5300000
	FST	X	
	FSQ	X	+3510160+4410160
	FST	X	
	FMP	RESULT	+3510160+0710140
	FST	RESULT	
	FCA	X	+3510140+3010160
	FSB	FLOCNO1	
	EXT	COMCNO1	+0610210+3310230
	FAD	FLOCNO2	
	TMI	LOOP	+0410240+5110050
	FCA	RESULT	
NORMAL	TRA	+0	+3010140+5700000
RESULT	DECIMAL	(-0)	-0000000-0000000
			-0000000-0000000
X	DECIMAL	(-0)	-0000000-0000000
			-0000000-0000000
FIXCNO1	DECIMAL	(+1+39)	+0000000-0000001
FLOCNO1	DECIMAL	(+1.0)	+4000000-0000000
			+0000000-0000001
COMCNO1	COMMAND	(-7777771+7777771)	-7777771+7777771
FLOCNO2	DECIMAL	(+0.0000000001)	+6700000-0000000
			-0000000-0000201
	END		